

xvfatに関するの議論

2005.05.20

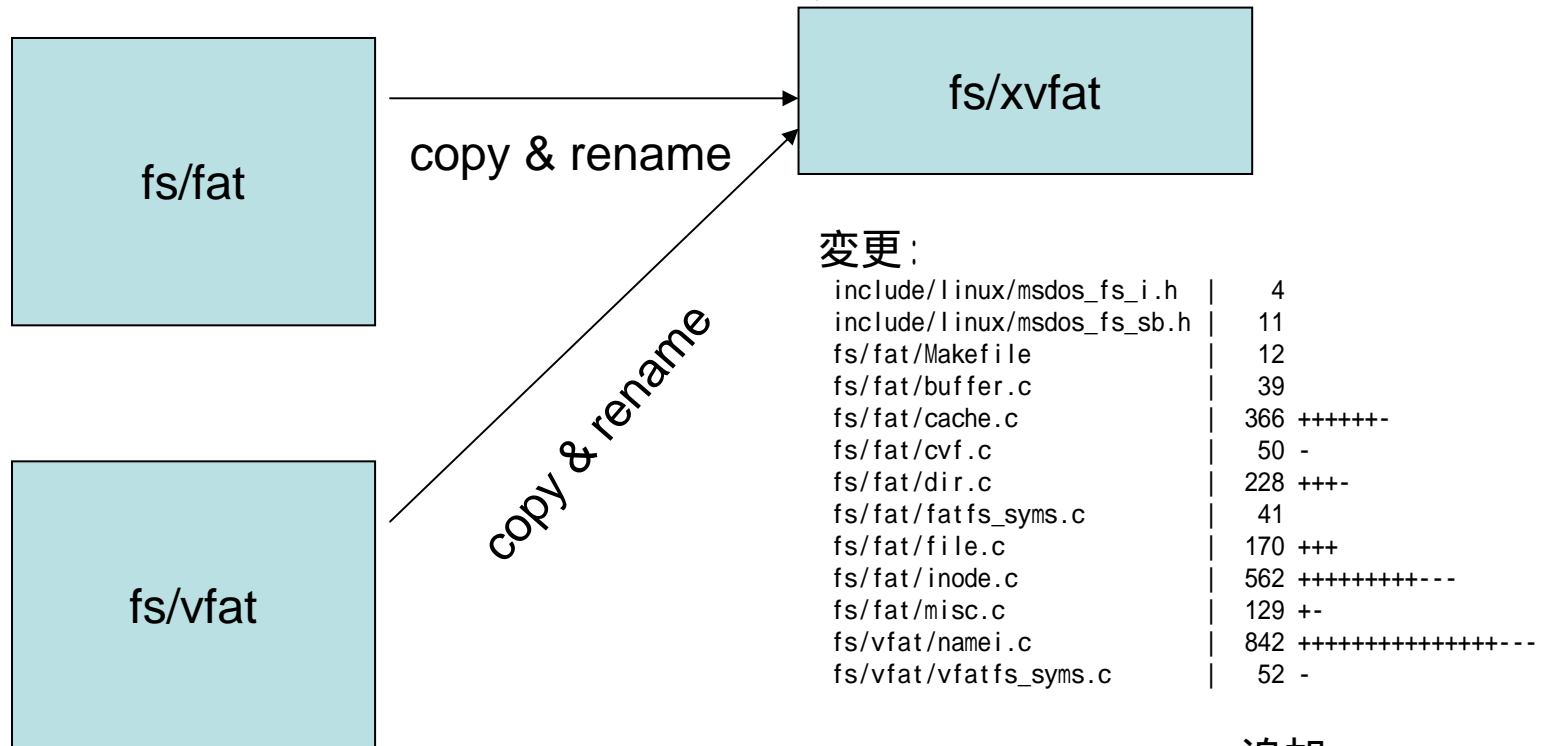
machida@sm.sony.co.jp

アジェンダ

- xvfatに関する紹介
- 抜去に関して
- Kernel 2.6へ向けて
 - ポーティング
 - コミュニティへのフィードバック

現在のsource code構成

- 従来のfatと共存できるように別物として構成



このほかにも、
エレベータ関数の変更あり
(driversの下)

追加:

2020	fs/xvfat/fwrq.c
1015	fs/xvfat/rdchk.c
610	fs/xvfat/tz.c
50	include/linux/rdchk.h
41	include/linux/fat_tz.h
224	include/linux/xvfat_fs.h
245	include/linux/fwrq.h

xvfatに関する紹介

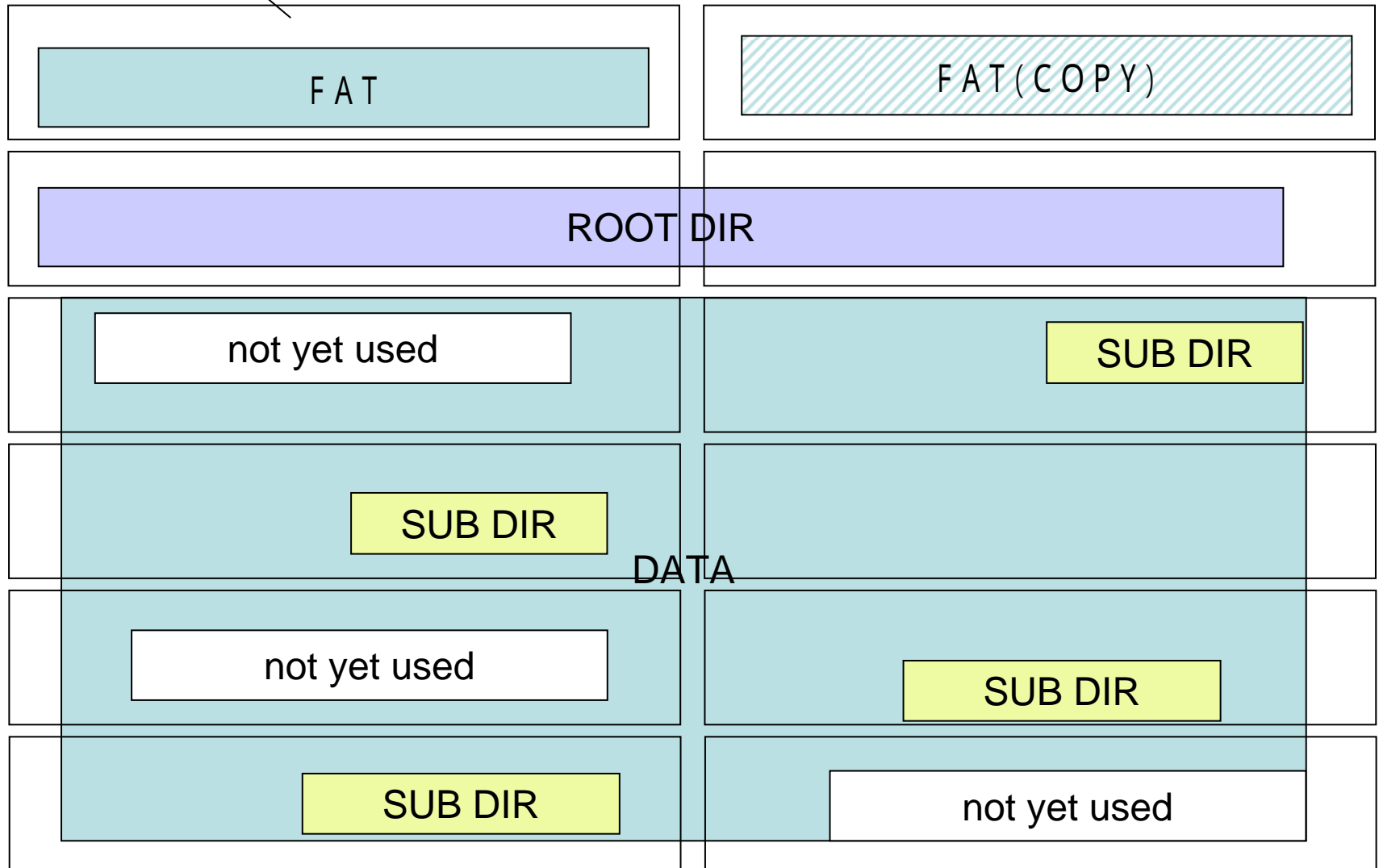
- 機能の紹介資料(以前のおさらい)
 - 20050325-xvfat-jpn.pdf
- 拡張の追加など
 - ファイルシステムのioctl
 - mediaの状態変化取得にカウンタを増設
 - 2bit のstateだけなので、本当に変わったかを確認するため
 - 日本語サポート
 - 実際にはunicode部分のfixもある

抜去対応

- 抜去検出と通知
 - mediaの抜去 pollingでドライバへ問い合わせし検出
 - deviceの抜去 deviceへの操作のエラーを記憶
 - Deviceが抜かれたときの挙動はドライバに依存
 - デバイスごと抜かれたことへの対応には弱い
- 抜かれたあとは、system call はエラーで返る
 - 操作ごとに毎回チェック
- 書き込み順序を保存(EraseBlockを超えて入れ替わらない)
- 同期書き込み
- EraseBlock単位でまとめて書き込まれるように調整
 - FATがerase block内になることが重要
 - erase block内は一気にかけると仮定して、トランザクション管理

FATの構成とEraseBlock

EraseBlock



例:create()

- 一度 I/Oエラーなら、以降はスキップ
- トランザクション開始
- ディレクトリへのエントリ追加を記録
- トランザクション終了
- FATの更新(開放以外)は直ぐに書き込み

例:ファイルへの書き込み

- 一度 I/Oエラーなら、以降はスキップ
- トランザクション開始
- データ書き込み
- ディレクトリ、FATの更新は直ぐに書き込み
- トランザクション終了

Kernel 2.6への対応(1/3)

- コミュニティーへのフィードバックをしないと
 - 細かい単位でパッチを分ける
 - 既存の仕組みをそのまま/改造して利用
- まずは、現状の2.4.20ベースでパッチ機能毎に分離
 - その後、各機能ごとに検討を進める
- 全体構成
 - オリジナルのfs/fat, fs/vfatを利用するようにできるか？

Kernel 2.6への対応(2/3)

- Erase Block単位でのトランザクション管理
 - 2.4.20
 - 独自実装
 - EraseBlockサイズは、MS用パラメータテーブル用意
 - 2.6での実装
 - journalling-API (jbd)が利用できるか？
 - Documentation/DocBook/journal-api.tmpl からdocは生成可能
 - RAM領域をexternal journalとして利用
 - Journal化も可能かも
 - EraseBlockサイズ ドライバのパラメータを利用？
 - Documentation/block/biodoc.txt
 - » max_seg_size は適当？
 - クラスタアロケーションも EraseBlockを考慮しては？
 - たとえば、先頭から空きが連続してあるEraseBlockを優先的につかうとか...

Kernel 2.6への対応(3/3)

- 挿抜検出/対応

- HOTPLUGを利用？
- 2.6ではll_rw_block.cにtime out処理が追加されている

- 書き込み順序制御

- 2.4
 - EraseBlock内はReorder可能と過程
 - 前記のトランザクション実装と独自エレベータ関数内で制御
- 2.6
 - EraseBlock内でのリオーダはOKとするか、しないかを選択する？
 - シーク時間の考慮をON/OFF できる
 - block device layer の queueのパラメータだけ制御可能か？
 - NO: バウンダリがあるのでだめなはず、elevator側で対処するしかないだろう

想定機器

- 内蔵非リムーバブル
 - USB clientになるデバイスで内蔵ストレージをPCからマウント (USB Mass storageとして公開)
 - Flash Memoryなど
- リムーバブル
 - USB hostになるデバイスで、メモリカードやデジカメ等をUSB Massとしてマウント
 - バス直結型(PCMCIAなど)メモリカードリーダー
- HDDは？

議論

- jbdは使えそうだけど、変更は必要そう
 - Dataの吐き出しのタイミング制御は自分でやる必要があるのでは？
 - JFSはkjournaldを使わずに自前で daemonを起こしているようだ
- そもそもなぜFATかを理解してもらうのが大変
 - 目的とか、前提を明確にして議論しないと話がかみ合わない可能性がある
- そもそもEraseBlock単位でよいのか？
 - 個別WriteBlockの書き込みの方が TransLayerがうまく扱えるのではないか
 - EraseBlock単位の読み込み書き込みは、バーストできることもあるはずなので有利なことが多いかも
- 下位にTransLayerと協調して動くべき
 - エラー時にEraseBlock内を全てあきらめるのではなくて、WriteBlock単位で読み込んで復旧するとか
 - エラー時に TransLayer側のキャッシュ等のメモリ内の記録を使ってデータの復旧を行うとか
- Journal化できるとよいかも
 - removable だと 抜去 PCでマウント、書き込みで戻すと logとデータ本体が一致しなくなるので注意が必要
 - どんなメリットがあるか？ 内蔵の場合と、removableでは違うのでは
- HDDとUSB Mass Storage、内蔵フラッシュメモリをサポートする際の差は？
 - シーク時間の考慮、ソフト側でのTransLayer有り無し、EraseBlock長、WriteBlock長など